IN THE CLAIMS

This listing of Claims shall replace all prior versions, and listings, of claims in the application:

1. (currently amended) A boot method for ~~an In-Circuit Emulation system having a microcontroller operating in lock-step synchronization with a virtual controller~~ synchronizing a microcontroller and a virtual microcontroller of an In-Circuit Emulation system in lock-step, comprising:

in the microcontroller, executing a set of boot code to ~~substantially~~ carry out initialization;

in the virtual microcontroller, executing a set of timing code to enable ~~the~~ a lock-step synchronization, wherein the set of timing code is a dummy code timed to take the same number of clock cycles as the microcontroller uses to execute the set of boot code, and wherein ~~at least one portion of said~~ the set of timing code is different from ~~said~~ the set of boot code, and wherein the set of boot code is stored within the microcontroller and ~~at least one portion of~~ the set of boot code is inaccessible to the virtual microcontroller; and

simultaneously halting both the microcontroller and the virtual microcontroller.

2. (original) The method according to Claim 1, further comprising copying register contents from the microcontroller to corresponding registers in the virtual microcontroller.

3. (canceled)

4. (currently amended) The method according to Claim 1, wherein after the executing of the boot code, the microcontroller branches to an assembly instruction line 0; and wherein after executing the timing code, the virtual microcontroller branches to ~~said~~ the assembly instruction line 0.

5. (previously presented) The method according to Claim 1, wherein prior to the executing of the boot code, and prior to executing the timing code, a break is set at an assembly instruction line 0.

6. (previously presented) The method according to Claim 1, wherein the boot code comprises protected initialization code that is not accessible to the In-Circuit Emulation system.

7. (currently amended) The method according to Claim 1, further comprising:

prior to the executing of the boot code, and prior to executing the timing code, setting a break at an assembly instruction line 0; and

wherein after the executing of the boot code the microcontroller branches to ~~said~~ the assembly instruction line 0; and

wherein after executing the timing code, the virtual microcontroller branches to ~~said~~ the assembly instruction line 0.

8. (currently amended) The method according to Claim 1, further comprising:

prior to the executing of the boot code, and prior to executing the timing code, setting a break at an assembly instruction line 0;

wherein after the executing of the boot code, the microcontroller branches to ~~said~~ the assembly instruction line 0; and wherein after executing the timing code, the virtual microcontroller branches to ~~said~~ the assembly instruction line 0;

copying register contents from the microcontroller to corresponding registers in the virtual microcontroller;

copying memory contents from the microcontroller to corresponding memory in the virtual microcontroller;

wherein after the executing of the boot code, the microcontroller branches to ~~said~~ the assembly instruction line 0; and

wherein after executing the timing code, the virtual microcontroller branches to ~~said~~ the assembly instruction line 0.

9. (currently amended) The method according to Claim 8, further comprising removing the break at the assembly line ~~zero~~ 0 after copying the register contents and copying the memory contents.

10. (currently amended) A boot method for ~~an In-Circuit Emulation system having a microcontroller operating in lock-step synchronization with a virtual controller~~ synchronizing a microcontroller and a virtual microcontroller of an In-Circuit Emulation system in lock-step, comprising:

resetting the microcontroller and the virtual microcontroller to a halt state;

setting a break at an assembly instruction line 0;

in the microcontroller, executing a set of boot code to ~~substantially~~ carry out initialization;

in the virtual microcontroller, executing a set of timing code to enable ~~the~~ a lock-step synchronization, wherein the set of timing code is a dummy code timed to take the same number of clock cycles as the microcontroller uses to execute the set of boot code, and wherein ~~at least one portion of said~~ the set of timing code is different from ~~said~~ the set of boot code, and wherein the set of boot code is stored within the microcontroller and ~~at least one portion of~~ the set of boot code is inaccessible to the virtual microcontroller;

simultaneously halting both the microcontroller and the virtual microcontroller by branching to ~~said~~ the assembly instruction line 0;

copying register contents from the microcontroller to corresponding registers in the virtual microcontroller;

copying memory contents from the microcontroller to corresponding memory in the virtual microcontroller; and

removing the break at ~~said~~ the assembly line 0 after copying the register contents and copying the memory contents.

11. (canceled)

12. (currently amended) A boot method for ~~an In-Circuit Emulation system having a device operating under test operating in lock-step synchronization with a virtual processor~~ synchronizing a tested device and a virtual processor of an In-Circuit Emulation system in lock-step, comprising:

in the tested device ~~under test~~, executing a set of boot code to ~~substantially~~ carry out initialization;

in the virtual processor, executing a set of timing code to enable ~~the~~ a lock-step synchronization, wherein the timing code is a dummy code timed to take the same number of clock cycles as the tested device ~~under test~~ uses to execute the set of boot code, and wherein at least one portion of ~~said~~ the set of timing code is different from ~~said~~ the set of boot code, and wherein the set of boot code is stored within the tested device ~~under test~~ and ~~at least one portion of~~ the set of boot code is inaccessible to the virtual processor; and

simultaneously halting both the tested device ~~under test~~ and the virtual processor.

13. (canceled)

14. (currently amended) The method according to Claim 12, further comprising copying memory contents from memory coupled to the tested device ~~under test~~ to corresponding memory coupled to the virtual processor.

15. (currently amended) The method according to Claim 12, wherein after the executing of the boot code, the tested device ~~under test~~ branches to an assembly instruction line 0; and wherein after executing the timing code, the virtual processor branches to ~~said~~ the assembly instruction line 0.

16. (previously presented) The method according to Claim 12, wherein prior to the executing of the boot code, and prior to executing the timing code, a break is set at an assembly instruction line 0.

17. (previously presented) The method according to Claim 12, wherein the boot code comprises protected initialization code that is not accessible to the In-Circuit Emulation system.

18. (currently amended) The method according to Claim 12, further comprising:

prior to the executing of the boot code, and prior to executing the timing code, setting a break at an assembly instruction line 0; and

wherein after the executing of the boot code, the tested device ~~under test~~ branches to ~~said~~ the assembly instruction line 0; and

wherein after executing the timing code, the virtual processor branches to ~~said~~ the assembly instruction line 0.

19. (currently amended) The method according to Claim 12, further comprising:

prior to the executing of the boot code, and prior to executing the timing code, setting a break at an assembly instruction line 0;

wherein after the executing of the boot code, the tested device ~~under test~~ branches to ~~said~~ the assembly instruction line 0; and wherein after executing the timing code, the virtual processor branches to ~~said~~ the assembly instruction line 0;

copying register contents from the tested device ~~under test~~ to corresponding registers in the virtual processor;

copying memory contents from the tested device ~~under test~~ to corresponding memory in the virtual processor;

wherein after the executing of the boot code, the tested device ~~under test~~ branches to ~~said~~ the assembly instruction line 0; and

wherein after executing the timing code, the virtual processor branches to ~~said~~ the assembly instruction line 0.

20. (original) The method according to Claim 19, further comprising removing the break at assembly line zero after copying the register contents and copying the memory contents.

21. (original) The method according to Claim 12, wherein the virtual processor is implemented in a field programmable gate array.

22. (currently amended) The method according to Claim 1, wherein ~~said~~ the set of boot code comprises proprietary information, wherein ~~said~~ the proprietary information comprises serial numbers, passwords, and algorithms.

23. (currently amended) The method according to Claim 1, wherein at least one portion of the boot code is inaccessible to the virtual microcontroller by being stored internally in ~~said~~ the microcontroller.